OpenRules Decision Model for
DMCommunity June-2024 Challenge "Smart Marriages"
(without Java)

Problem

This challenge deals with the famous stable marriage problem formulated as follows:

"Given n men and n women, where each person has ranked all members of the opposite sex in order of preference, marry the men and women together such that there are no two people of the opposite sex who would both rather have each other than their current partners. When there are no such pairs of people, the set of marriages is deemed stable."

Here is an example of the problem:

How Men Rank Women:

	Alice	Barbara	Claire	Doris	Elsie
Adam	5	1	2	4	3
Bob	4	1	3	2	5
Charlie	5	3	2	4	1
Dave	1	5	4	3	2
Edgar	4	3	2	1	5

How Women Rank Men:

	Adam	Bob	Charlie	Dave	Edgar
Alice	1	2	4	3	5
Barbara	3	5	1	2	4
Claire	5	4	2	1	3
Doris	1	4	3	2	5
Elsie	4	2	3	5	1

I've already submitted a solution using OpenRules Decision Manager with RuleSolver. I offered two different implementation approaches and both of them contain technical parts implemented in a Java class with direct calls of JSR-331 (Constraint Programming API). Still, I wanted to do the same without Java providing a solution more oriented to a business person. Below I describe such a solution that implements Approach 1 but without Java. The complete solution can be found in the file StableMarriageExcel.xls.

Solution

As in Approach 1, I didn't need to implement complex marriage stability constraints. Instead, I implemented simple constraints that state:

"Each person should marry one and only one person of the opposite gender".

Then I defined preferences for all pairs "man-woman" and "woman-man" and calculated the sum of all these preferences. It became my optimization objective. When I minimized this objective, I received a solution with the most top preferences being satisfied.

Business Part

I started with the creation of a business glossary:

Glossary glossary			
Variables	**Business Concept**	**Attribute**	**Type**
Men	**Marriages**	men	Person[]
Women		women	Person[]
Name	**Person**	name	String
Preferences		preferences	int[]
Assigned Pairs	**Results**	assignedPairs	String[]

Then I created test data for two instances of the problem of sizes 5 and 6:

DecisionData Person men5	
Person Name	**Person Preferences**
Adam	5,1,2,4,3
Bob	4,1,3,2,5
Charlie	5,3,2,4,1
Dave	1,5,4,3,2
Edgar	4,3,2,1,5

DecisionData Person women5	
Person Name	**Person Preferences**
Alice	1,2,4,3,5
Barbara	3,5,1,2,4
Claire	5,4,2,1,3
Doris	1,4,3,2,5
Elsie	4,2,3,5,1

DecisionData Person men6	
Person Name	**Person Preferences**
Adam	1,2,4,5,6,3
Bob	4,2,5,3,1,6
Charlie	2,5,4,6,3,1
Dave	3,4,2,6,5,1
Edgar	6,3,2,1,5,4
Fred	4,6,3,2,1,5

DecisionData Person women6	
Person Name	**Person Preferences**
Alice	5,2,6,3,1,4
Barbara	6,5,1,2,4,3
Claire	3,2,1,4,5,6
Doris	2,4,1,3,5,6
Elsie	3,5,4,2,1,6
Fiona	5,1,6,4,2,3

I added these samples as test cases for my OpenRules-based decision model and added expected results from my previous solution:

DecisionTest testCases			
#	ActionDefine	ActionDefine	ActionExpect
Test	**Women**	**Men**	**Assigned Pairs**
1	women5	men5	Adam-Barbara Bob-Doris Charlie-Elsie Dave-Alice Edgar-Claire
2	women6	men6	Adam-Alice Bob-Barbara Charlie-Claire Dave-Fiona Edgar-Doris Fred-Elsie

To check that my test data satisfies the requirement “there should be the same number of men and women”, I added the following decision table:

Decision ValidateData			
Condition		Message	ActionExecute
Count of Men		ERROR	Rules
!=	Count of Women	"Number of men and women should be the same"	ACTION-TERMINATE

Solver Part

I used the following table to create all unknown decision variables:

Decision DefineAllVariables [for each Man in Men; for each Woman in Women]			
SolverDefineVariables			
Variable Name	**Method Name**	**Par 1**	**Par 2**
{{Name of Man}}-{{Name of Woman}}	"New Variable"	"0"	"1"

They all are constrained variables with possible values of 0 or 1 and names like “Adam-Alice”, “Adam-Barbara”, etc.

To state that “*each man should marry one and only one woman*” I decided to organize a loop for each Man in the array of Men:

Decision PostMenConstraints [for each Man in Men]
ActionExecute
Decision Tables
DefineManVariables
DefineSumOfManVariables
PostManConstraints

This loop executes the following decision tables:

Decision DefineManVariables [for each Woman in Women]		
SolverDefineVariables		
Variable Name	**Method Name**	**Variable Name**
{{Name of Man}} Variables	"Add Variable"	{{Name of Man}}-{{Name of Woman}}

Decision DefineSumOfManVariables		
SolverDefineVariables		
Expression Name	**Method**	**Variables**
Sum of {{Name of Man}} Variables	"Sum"	{{Name of Man}} Variables

Decision PostManConstraints				
SolverPostConstraints				
Constraint Name	**Constraint Type**	**Par 1**	**Par 2**	**Par 3**
"Woman should marry one and only one man"	"Variable Operator Value"	Sum of {{Name of Man}} Variables	"="	"1"

Then I created similar 4 tables to state that *"each woman should marry one and only one man".* Here they are:

Decision PostWomenConstraints [for each Woman in Women]
ActionExecute
Decision Tables
DefineWomanVariables
DefineSumOfWomanVariables
PostWomanConstraints

Decision DefineWomanVariables [for each Man in Men]		
SolverDefineVariables		
Variable Name	**Method Name**	**Variable Name**
{{Name of Woman}} Variables	"Add Variable"	{{Name of Man}}-{{Name of Woman}}

Decision DefineSumOfWomanVariables		
SolverDefineVariables		
Expression Name	**Method**	**Variables**
Sum of {{Name of Woman}} Variables	"Sum"	{{Name of Woman}} Variables

Decision PostWomanConstraints				
SolverPostConstraints				
Constraint Name	**Constraint Type**	**Par 1**	**Par 2**	**Par 3**
"Man should marry one and only one woman"	"Variable Operator Value"	Sum of {{Name of Woman}} Variables	"="	"1"

To define an array “Men Preferences” I organized the following loop:

Decision DefineManPreferenceVariables [for each Man in Men; ManPreferences = Preferences of Man]			
SolverDefineVariables			
Variable Name	**Method Name**	**Par 1**	**Par 2**
{{Name of Man}} Preferences	"ScalarProduct"	{{Name of Man}} Variables	"ManPreferences"
"Men Preferences"	"Add Variable"	{{Name of Man}} Preferences	

For each Man from the array Men it calculates an array of ManPreferences with integer values and then uses this array to define preference variables such as “Adam Preferences” which as a scalar product of Adam Variables and ManPreferences. The second rule in this table adds this just-created variable to the array “Men Preferences”.

Similarly, I organized the following loop to define an array “Women Preferences”:

Decision DefineWomanPreferenceVariables [for each Woman in Women; WomanPreferences = Preferences of Woman]			
SolverDefineVariables			
Variable Name	**Method Name**	**Par 1**	**Par 2**
{{Name of Woman}} Preferences	"ScalarProduct"	{{Name of Woman}} Variables	"WomanPreferences"
"Women Preferences"	"Add Variable"	{{Name of Woman}} Preferences	

The only remaining step was to summarize all men's and women's preferences. I did it in the following table in which I first defined two variables "Sum of Men Preferences" and "Sum of Women Preferences" and then defined "All Preferences" as a sum of these two variables:

Decision DefineAllPreferences				
SolverDefineVariables				
Expression Name	**Method**	**Variable**	**Oper**	**Variable**
"Sum of Men Preferences"	"Sum"	"Men Preferences"		
"Sum of Women Preferences"		"Women Preferences"		
"All Preferences"	"Variable Operator Variable"	"Sum of Men Preferences"	"+"	"Sum of Women Preferences"

Finally, as we usually do for Rule Solver, I needed to define two methods “Define” and “Solve” to complete my decision model. Here is the method “Define” that simply executes previously specified decision tables:

Decision Define
ActionExecute
Decision Tables
ValidateData
DefineAllVariables
PostMenConstraints
PostWomenConstraints
DefineManPreferenceVariables
DefineWomanPreferenceVariables
DefineAllPreferences

And here is the method “Solve” that sets our optimization objective “All Preferences” and uses the standard methods “Solver Minimize” and “SolverLogSolution” to minimize and print the optimal solution:

Decision Solve
ActionExecute
Actions
SetObjective
SolverMinimize
SolverLogSolution
AssignSolution

Decision SetObjective
SolverSetObjective
Objective
"All Preferences"

I wanted to save the found solution in the array “Assigned Pairs” defined in our Glossary as the decision model output. I did it using the following decision table:

Decision AssignSolution [for each Man in Men; for each Woman in Women]				
SolverSolutionCondition			Action	
Variable Name	Operator	Value	Assigned Pairs	
{{Name of Man}}-{{Name of Woman}}	=	1	Add	{{Name of Man}}-{{Name of Woman}}

When I executed my decision model I received the expected results for both test cases that correspond to the same results from a similar implementation but done in Java:

DecisionTest testCases			
#	ActionDefine	ActionDefine	ActionExpect
Test	Women	Men	Assigned Pairs
1	women5	men5	Adam-Barbara Bob-Doris Charlie-Elsie Dave-Alice Edgar-Claire
2	women6	men6	Adam-Alice Bob-Barbara Charlie-Claire Dave-Fiona Edgar-Doris Fred-Elsie

```
Optimal Solution with top satisfied preferences: 23
Adam(1) - Barbara(3)
Bob(2) - Doris(4)
Charlie(1) - Elsie(3)
Dave(1) - Alice(3)
Edgar(2) - Claire(3)
```

```
Optimal Solution with top satisfied preferences: 36
Adam(1) - Alice(5)
Bob(2) - Barbara(5)
Charlie(4) - Claire(1)
Dave(1) - Fiona(4)
Edgar(1) - Doris(5)
Fred(1) - Elsie(6)
```

The execution time was also the same and extremely fast (around 100 milliseconds per test case).